Pertemuan 7

File pada Java

Objektif :

1. Mahasiswa dapat memahami latar belakang penggunaan File
2. Mahasiswa dapat memahami tentang Java IO Stream
3. Mahasiswa dapat mengetahui tentang Class dan Method File pada Java
4. Mahasiswa dapat mengetahui macam-macam Operasi File
5. Mahasiswa dapat membuat program sederhana tentang File pada Java

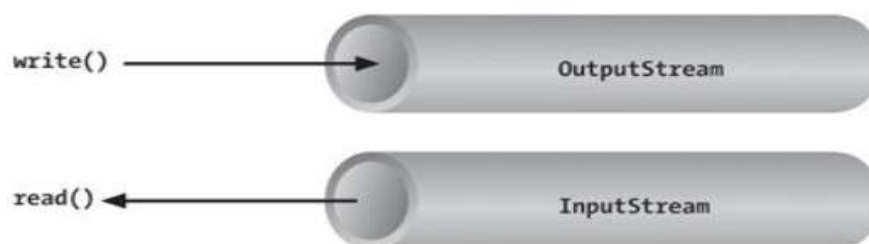
P7.1 Teori

1. Latar Belakang Penggunaan File

Data yang selama ini disimpan di dalam suatu variabel dan array bersifat sementara. Artinya, data tersebut akan hilang pada saat program berhenti. Komputer biasanya menggunakan fileuntuk penyimpanan yang bersifat menetap, walaupun program yang membuat data tersebut dihentikan.

Pemrosesan file merupakan salah satu kemampuan penting dari suatu bahasa pemrograman, agar mampu menghasilkan aplikasi komersil yang membutuhkan penyimpanan data secara menetap dalam jumlah yang besar.

2. Java IO Stream

Stream merupakan dasar operasi input-output (I/O) dalam Java yang menggunakan package java.io sebagai package utama. ***Stream*** adalah representasi abstrak dari input dan output device, dimana aliran bytes akan ditransfer seperti file dalam harddisk, file pada sistem remote atau printer. Kita dapat membaca data dari input stream, yang dapat berupa file, keyboard atau komputer remote. Sedangkan untuk operasi penulisan berarti menulis data pada output stream. Package java.io mendukung dua tipe stream, yaitu ***binari*** dan ***karakter*** stream. ***Binari*** merupakan data berupa bit atau data binari, sedangkan ***karakter*** adalah tipe khusus untuk pembacaan dan penulisan teks/karakter.

Input Stream Subclass-subclass dari inputStream adalah : AudioInputStream, ByteArrayInputStream, FileInputStream, FilterInputStream, PipedInputStream, SequenceInputStream, dan StringBufferInputStream. Dua method utama dari InputStream adalah :

• Read

Method ini digunakan untuk membaca stream.

• Close

Method ini digunakan untuk menutup koneksi input stream.

Output Stream Subclass-subclass dari outputStream adalah :

- ByteArrayOutputStream : digunakan untuk menuliskan stream menjadi byte array.
- FileOutputStream : digunakan untuk menulis pada file
- FilterOutputStream : merupakan superclass dari subclass-subclass seperti DataOutputStream, BufferOutputStream, PrintStream, CheckedOutputStream
- ObjectOutputStream : digunakan untuk menuliskan objek pada OutputStream.
- PipedOutputStream : digunakan untuk menjadi output dari PipedInputStream.

Sebagian method-method OutputStream adalah :

- Voidclose()

  Menutup output stream yang aktif dan melepaskan sumber daya terkait dengan stream tersebut.
- Void flush()

  Melakukan flush output stream dan memaksa semua byte buffer untuk dituliskan keluar
- Void write(byte[] b)

  Menulis sebanyak b.length dari byte array ke output stream
- Void write(byte[] b, int off, int len)

  Menuliskan sebanyak len byte dari byte array b dimulai dari index off

Program Java melakukan pemrosesan file dengan menggunakan class-class dari **packagejava.io**. Package java.io ini berisikan class-class streams seperti:

1. **FileInputStream**

   Untuk input berupa byte dari suatu file
2. **FileOutputStream**

   Untuk output berupa byte kepada suatu file
3. **FileReader**

   Untuk input berupa karakter dari suatu file
4. **FileWriter**

   Untuk output berupa karakter kepada suatu file

3. Class dan Method File Pada Java

Class File adalah kunci dalam pemrosesan File atau Direktori. Objek File merepresentasikan single file atau directory. Class File berguna untuk mengambil

informasi mengenai suatu file atau direktori dari disk. Constructor class File tediri dari :

a. public File(String name)
b. public File(String pathToName, String name)
c. public File(File directory, String name)
d. public File(URI uri)

Ada 3 cara membuat objek File, yaitu :

1. Menggunakan objek String sebagai argument yang menginformasikan path untuk file atau direktori. Contoh :

   File direktori = new File("c:\\my documents\\java\\");

   File fileku = new File("c:\\my documents\\java\\dokumen.txt");

2. Menggunakan dua langkah dimana yang pertama untuk mendefinisikan direktori dan yang kedua untuk file. Contoh :

   File dirku = new File("c:\\my documents\\java");

   File filenya = new File(dirku, "dokumennya.txt");

3. Menggunakan dua argument dimana yang pertama adalah argument String yang mendefinisikan direktori, dan yang kedua adalah argument String yang mendefinisikan nama file. Contoh :

   File filesaya = new File("c:\\my documents\\java\\","dokumennya.txt");

<u>Method dalam Class File</u>

Selain mendefinisikan direktori atau file, objek File juga dapat digunakan untuk mendapatkan informasi file melalui method-method yang ada dalam objek File.

Method	Keterangan
exists()	Mengembalikan nilai true apabila file atau direktori itu ada
isDirectory()	Mengecek apakah objek file menunjuk pada direktori atau file
getName()	Mendapatkan nama file atau direktori dalam string(tanpa path) dari objek File
getPath()	Mendapatkan path dalam String dari objek File,termasuk nama File / direktori
getAbsolutePath()	Mendapatkan path absolute dari direktori / file yang direferensi oleh objek file
list()	Bila objek File mewakili direktori maka akan mengembalikan array String yang mengandung nama dari isi direktori. Bila objek File merupakan file maka akan mengembalikan nilai null
listFiles()	Bila objek file berupa direktori maka akan mengembalikan objek File yang ada dalam direktori
length()	Mengembalikan nilai bertipe long yang merupakan panjang bytes dari file yang diwakili oleh objek File, bila berupa direktori akan mengembalikan nilai 0
lastModified()	Mengembalikan nilai long yang mewakili waktu terakhir objek File terakhir kali dimodifikasi.

4. Contoh Teknik Operasi File

Berikut ini adalah macam-macam teknik operasi file pada Java antara lain :

1. Membuat File
2. Menampilkan nama File dan Direktori
3. Me-rename File
4. Menghapus File
5. Menghapus non-empty Direktori

Langkah-langkah yang dilakukan dalam teknik operasi file yaitu :

1. Membuat Objek File

   Untuk membuat object File, kita cukup memanggil salah satu constructor-nya. Contoh:

```
String path = "c:\\logs\\hits.log";
File f = newFile(path);
if(!f.exists())
System.out.println("The input file does not exist!");
```

2. Menampilkan Isi Direktori

   Kode berikut ini menampilkan nama-nama file yang ada dalam suatu direktori:

```
File dir = newFile(path);
if(dir.isDirectory()) {
File[] files = dir.listFiles();
```

```
for(File f : files)
System.out.println(f.getName());
}
```

3. Menampilkan hanya file saja, tidak menampilkan subdirektori maupun hidden files.

```
File dir = new File(path);
if (dir.isDirectory()) {
File[] files = dir.listFiles();
for (File f : files) {
if (f.isFile() && !f.isHidden())
System.out.println(f.getName());
}
}
```

4. Merename File

   Kode berikut mengubah nama sebuah file dari: hits.logmenjadi savedhits.log:

```
File f = newFile("hits.log");
if (f.renameTo(newFile("savedhits.log")))
System.out.println("File renamed.");
else
System.out.println("File not renamed.");
```

5. Menghapus File

   Kode berikut digunakan untuk menghapus file :

```
File f = newFile("hits.log");
if (f.delete())
System.out.println("File deleted.");
else
System.out.println("File not deleted.");
```

Untuk menghapus sebuah folder termasuk file dan subdirektori di dalamnya, kita cukup memanggil method tersebut:

```
deleteFile(newFile("nama_folder_yg_dihapus");
```

<u>Baca dan Tulis File</u>

- Membaca isi karakter File
- Menuliskan karakter ke dalam File

BufferReader

- Class BufferedReaderdapat “membungkus” class FileReaderuntuk menyediakan proses input yang lebih efisien.
- Class ini menambahkan suatu “buffer” kepada input stream sehingga input tersebut dibaca dalam “potongan besar” dari harddisk daripada byte-per-byte.
- Hal ini menghasilkan peningkatan performance.
- Class BufferedReader juga memungkinkan kita untuk membaca data secara per-karakter atau per-baris.

Langkah-langkah:

- Buat object File
- Buat objek FileReader
- Buat objek BufferedReader

Contoh membuat objek BufferedReader untuk membaca file “movie.txt”:

```
File f = newFile("movies.txt");
BufferedReader in = newBufferedReader(new FileReader(f));
```

Read & ReadLine

Kita dapat menggunakan methods “read” dan “readLine” untuk membaca isi objek BufferedReader.

int read()

- Membaca satu karakter dari file dan me-return suatu angka.
- Menghasilkan -1 apabila end-of-file telah dicapai.
- Throws IOException

String readLine()

- Membaca satu baris dan me-return-nya sebagai String.
- Me-return nullapabila end-of-file telah dicapai.
- Throws IOException

Menuliskan String ke File Teks

FileWriter

FileWriter merupakan subclass dari OutputStreamWriter dimana class OutputStreamWriter adalah subclass dari class abstrak Writer.

Class Writer memiliki Konstruktor yang umum seperti berikut :

· FileWriter(File objekfile);

· FilWriter(String pathkefile);

· FileWriter(String pathkefile, Boolean append);

Contoh penggunaan :

File inifile = (pathdirektori, namafile);

FileWriter outputnya = new FileWriter(inifile);

PrintWriter

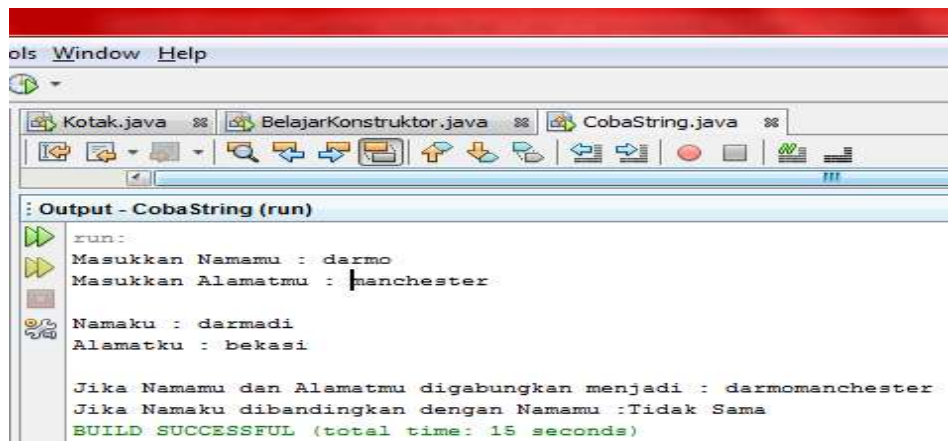
PrintWriter merupakan subclass dari class abstrak Writer yang digunakan melakukan output dari berbagai macam tipe data yang kemudian dikonversi ke bentuk karakter. Penggunaan PrintWriter dengan FileWriter :

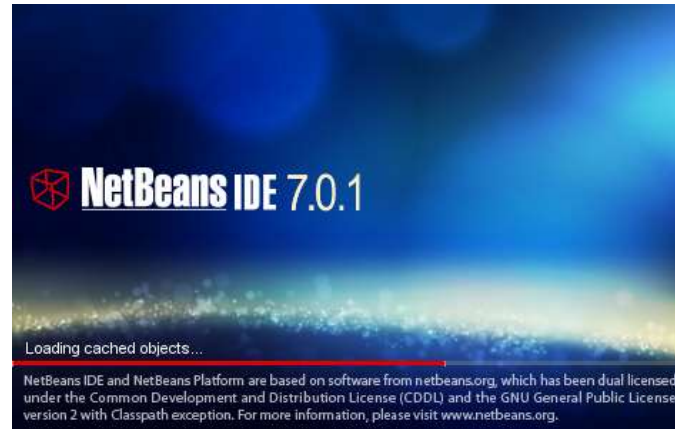
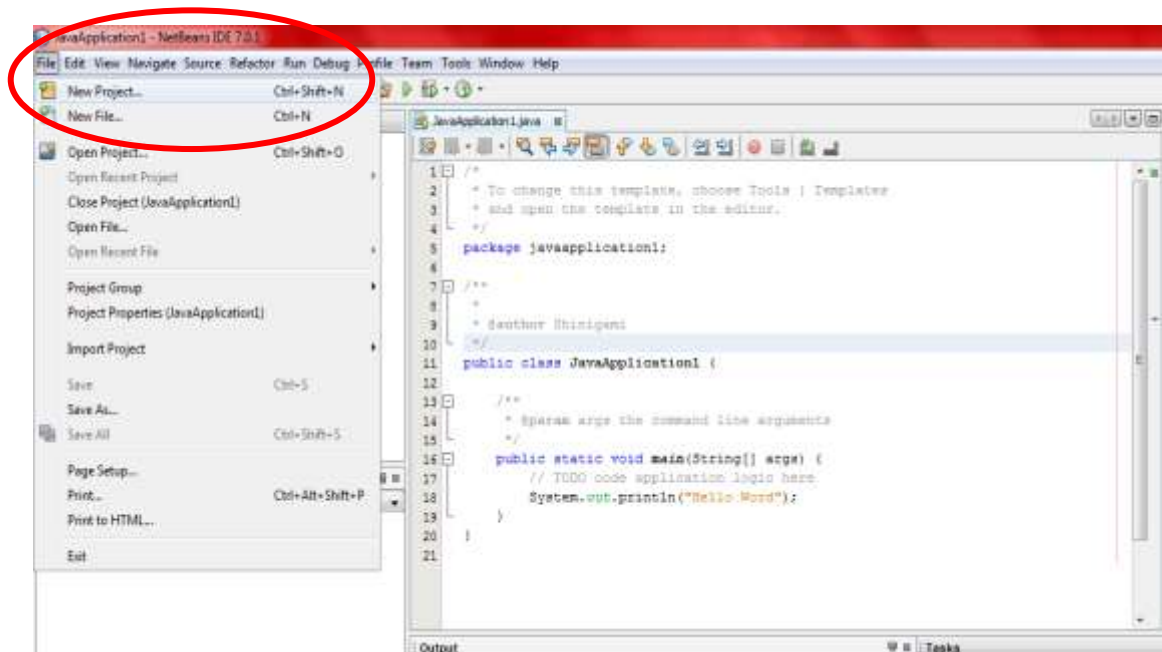
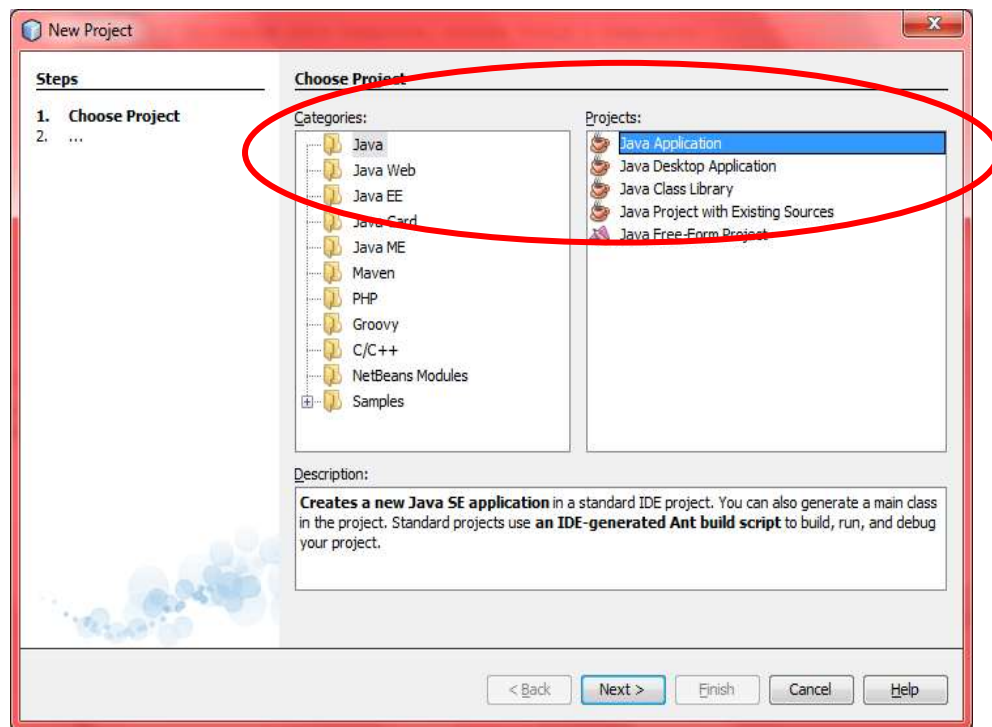
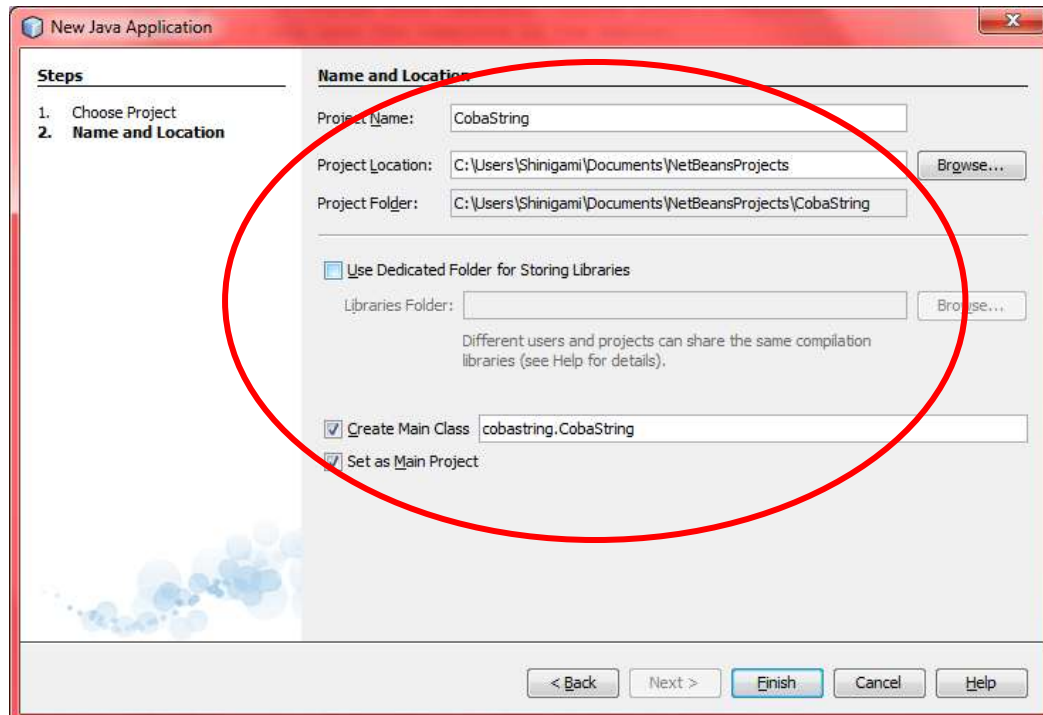
PrintWriter fileoutput = new PrintWriter (

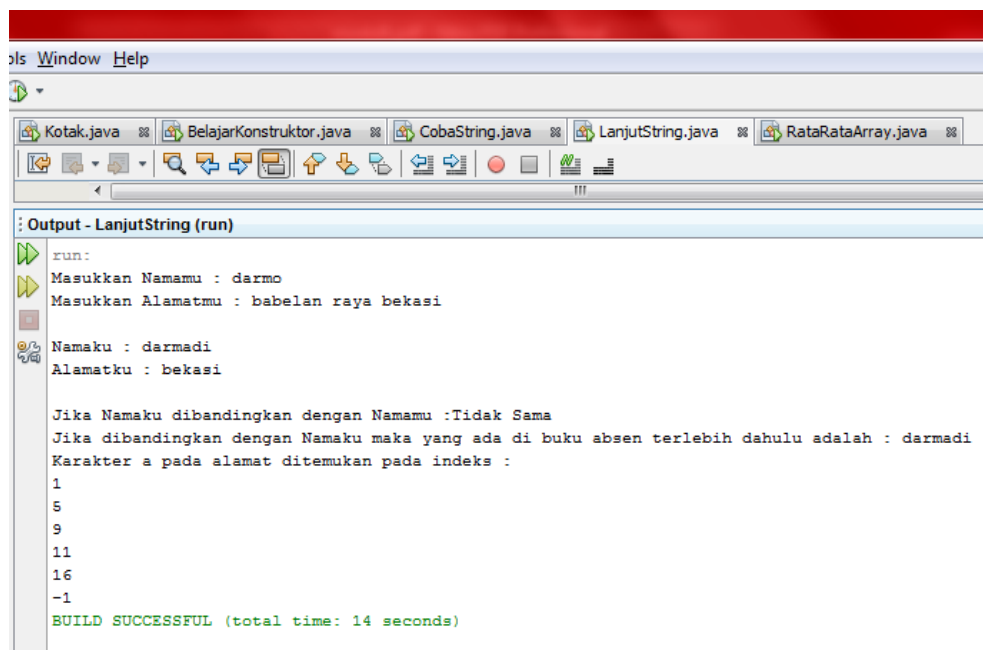
New FileWriter(File objekfile);

);

P7.2 Contoh Kasus

Buat program sederhana tentang menggabungkan dan membandingkan string dengan hasil output sebagai berikut:

Langkah-langkah Pengerjaan:

1. Jalankan Netbeans

2. Buat file project baru dengan memilih menu File – New Project, atau dengan menggunakan hotkey Ctrl+Shift+N.

3. Pilih jenis project yang akan dibuat (Java – Java Application)

4. Nama Project beserta nama classnya adalah CobaString

5. Ketikkan kode program di bawah ini pada code editor

```
package cobastring;

import java.io.*;
public class CobaString {

    public static void main(String[] args) throws Exception{
        // TODO code application logic here
        DataInputStream masuk = new DataInputStream(System.in);
        String nama,alamat;

        System.out.print("Masukkan Namamu : ");
        nama = masuk.readLine();
        System.out.print("Masukkan Alamatmu : ");
        alamat = masuk.readLine();

        StringCoba coba = new StringCoba(nama,alamat);
        System.out.println();
        System.out.println("Namaku : "+coba.namaku);
        System.out.println("Alamatku : "+coba.alamatku);
        System.out.println();
        System.out.println("Jika Namamu dan Alamatmu digabungkan menjadi :
"+coba.gabungString());

        // membandingkan nilai dari string namaku yang telah dideklarasikan dengan
string nama yang diinput user
        System.out.println("Jika Namaku dibandingkan dengan Namamu
:"+(coba.namaku.equalsIgnoreCase(nama)?"Sama":"Tidak Sama"));
    }
}
class StringCoba{
    String Nama1,Alamat1, gabung;
        String namaku = "darmadi";
        String alamatku = "bekasi";

    public StringCoba(String a, String b){
        Nama1 = a;
        Alamat1 = b;
    }
    public String gabungString(){
        gabung = Nama1 + Alamat1;   /* menggabungkan nilai string Nama1 dengan
                                     * string Alamat1 ke dalam string gabung */
        return gabung;
    } }
```

6. Build project tersebut dengan memilih menu Run – Build Main Project, atau dengan menggunakan hotkey F11.

7. Jika tidak ada kesalahan (BUILD SUCCESSFUL), jalankan project tersebut dengan memilh menu Run – Run Main Project, atau dengan menggunakan hotkey F6.

P7.3 Latihan

Buat program sederhana untuk mengurutkan string dan mencari karakter pada string. Output program tersebut:

Jawaban:

1. Jalankan Netbeans Anda
2. Lakukan langkah-langkah pengerjaan seperti contoh kasus sebelumnya.
3. Pada code editor Netbeans, ketikkan program berikut:

```
package lanjutstring;

import java.io.*;
public class LanjutString {

    public static void main(String[] args) throws Exception{
        // TODO code application logic here
        DataInputStream masuk = new DataInputStream(System.in);
        String nama,alamat;

        System.out.print("Masukkan Namamu : ");
        nama = masuk.readLine();
        System.out.print("Masukkan Alamatmu : ");
        alamat = masuk.readLine();
```

```
        StringCoba coba = new StringCoba(nama,alamat);
        System.out.println();
        System.out.println("Namaku : "+coba.namaku);
        System.out.println("Alamatku : "+coba.alamatku);
        System.out.println();
        System.out.println("Jika Namaku dibandingkan dengan Namamu
:"+(coba.namaku.equalsIgnoreCase(nama)?"Sama":"Tidak Sama"));
        System.out.println("Jika dibandingkan dengan Namaku maka yang ada di buku
absen terlebih dahulu adalah : "+coba.urutString());
        System.out.println("Karakter a pada alamat ditemukan pada indeks : ");
        int indeks[] = new int[alamat.length()];
        for(int i=0;i<alamat.length();i++){
            indeks[i] = alamat.indexOf('a',i);
            }
        int k=0, jumlah=0, j;
        while(k<(alamat.length()-1)){
            if(indeks[k]==-1)
                break;
            j=k+1;
            if(indeks[j]!=indeks[k]){
                jumlah++;
                if(jumlah==1)
                    System.out.println(indeks[k]);
                    System.out.println(indeks[j]);
                    k=j;
            }
            else{k++;}
        }
    }
}

class StringCoba{
    String Nama1,Alamat1, gabung, temp;
        String namaku = "darmadi";
        String alamatku = "bekasi";

    public StringCoba(String a, String b){
        Nama1 = a;
        Alamat1 = b;
    }

    public String gabungString(){
        gabung = Nama1 + Alamat1;
        return gabung;
    }

    public String urutString(){
        if(namaku.compareTo(Nama1)>0){
            temp = Nama1;
            Nama1 = namaku;
            namaku = temp;
        }
        return namaku;
    }
}
```

P7.4 Daftar Pustaka

Ady Wicaksono, *Dasar-dasar Pemrograman Java*, PT. Elex Media Komputindo, Jakarta 2002.

Benny Hermawan, *Menguasai Java 2 Object Oriented Programming*, Andi, Yogyakarta, 2004.

Ginanjar Utama, *Berfikir Objek: Cara Efektif Menguasai Java*, 2003. http://ilmukomputer.com/berseri/ginanjar-java/index.php (26 Desember 2004)

Indrajani dan Martin, *Pemrograman Berorientasi Objek dengan Java*, PT. Elex Media Komputindo, Jakarta, 2004.